Read the collapsed comments in authController.js before proceeding.

What JWT actually is (conceptually)

JWT = JSON Web Token

But forget the fancy name. Think of JWT as:

A signed identity card that the server gives to the client after login

That's it.

Step 1: What problem JWT is solving (again, but clearer)

You already understood this part:

- Login request happens ✓
- Credentials are verified ✓
- Server says "you are valid" ✓
- **But HTTP forgets everything after the response** ❌

So we need a way for the **client to prove its identity again and again**.

JWT exists to solve **only this**.

Step 2: What JWT is NOT

Let's remove confusion first.

JWT is **NOT**:

- ❌ Authentication itself
- ❌ Authorization rules

- ❌ Encryption of data
- ❌ A database replacement
- ❌ A “login system”

JWT is just a **proof of identity**, nothing more.

Step 3: What JWT actually contains

A JWT is just **text** (a string), made of **three parts**:

header.payload.signature

Conceptually:

1️⃣ **Header**

“What algorithm was used to sign this token?”

Example idea:

- HS256 (HMAC)
- RS256 (RSA)

Nothing about the user here.

2️⃣ **Payload (VERY IMPORTANT)**

This is the **identity data**.

Typically contains:

- user id
- email

- issued time
- expiry time

Example conceptually:

```
{
  "userId": "abc123",
  "email": "test@gmail.com",
  "exp": 1700000000
}
```

⚠️ **Payload is NOT encrypted**
Anyone can read it.
That's why:

- ❌ Never store passwords
- ❌ Never store sensitive info

3️⃣ Signature (THE CORE SECURITY)

The signature proves:

"This token was created by *this server* and was not modified"

It is created using:

- payload
- header
- **secret key (only server knows)**

If payload is changed:

- signature becomes invalid
- token is rejected

This is why JWT is trusted.

Step 4: JWT flow (real-world analogy)

Think of JWT like a cinema wristband 🎟️

1. You buy a ticket (login)
2. Staff checks your identity
3. Staff puts a **wristband** on you (JWT)
4. Every time you enter a hall:
    - You show the wristband
    - They don’t recheck your ticket
    - They just verify it’s real

The wristband:

- is issued once
- is checked many times
- expires after some time

Step 5: How JWT is used in requests

After login:

- Server sends JWT to client
- Client stores it (usually memory / localStorage)

Every protected request:

Authorization: Bearer <JWT>

Server:

- verifies signature
- extracts user data
- allows or denies access

Step 6: IMPORTANT distinction (this clears your confusion)

🔹 **Authentication**

“Who are you?”

- Login
- Token verification
- Identity proof

JWT is used **here**.

🔹 **Authorization**

“What are you allowed to do?”

Examples:

- Can update this task?
- Is this task yours?
- Are you admin?

JWT only helps by telling **who you are**, not what you can do.